# The Elastic Lottery Ticket Hypothesis

**Xiaohan Chen** [1] **Yu Cheng** [2] **Shuohang Wang** [2] **Zhe Gan** [2] **Jingjing Liu** [2] **Zhangyang Wang** [1]

## Abstract

Lottery Ticket Hypothesis raises keen attention to identifying sparse trainable subnetworks, or winning tickets, at the initialization (or early stage) of training, which can be trained in isolation to achieve similar or even better performance compared to the full models. Despite many efforts being made, the most effective method to identify such winning tickets is still *Iterative Magnitude-based Pruning (IMP)*, which is computationally expensive and has to be run thoroughly for every different network. A natural question that comes in is: *can we "transform" the winning ticket found in one network to another with a different architecture, yielding a winning ticket for the latter at the beginning, without re-doing the expensive IMP?* Answering this question is not only practically relevant for efficient "once-for-all" winning ticket finding, but also theoretically appealing for uncovering inherently scalable sparse patterns in networks. We conduct extensive experiments on CIFAR-10 and ImageNet, and propose a variety of strategies to tweak the winning tickets found from different networks of the same model family (e.g., ResNets). Based on these results, we articulate the *Elastic Lottery Ticket Hypothesis* (**E-LTH**): by mindfully replicating (or dropping) and re-ordering layers for one network, its corresponding winning ticket could be stretched (or squeezed) into a subnetwork for another deeper (or shallower) network from the same family, whose performance is nearly the same competitive as the latter's winning ticket directly found by IMP. We have also thoroughly compared E-LTH with pruning-at-initialization and dynamic sparse training methods, and discuss the generalizability of E-LTH to different model families, layer types, and even across datasets. Our codes are publicly available at GitHub.

[1]Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA [2]Microsoft Corporation. Correspondence to: Xiaohan Chen <xiaohan.chen@utexas.edu>.

ArXiv preprint. Work in progress.

## 1. Introduction

*Lottery Ticket Hypothesis (LTH)* (Frankle & Carbin, 2019) suggests the existence of sparse subnetworks in over-parameterized neural networks at their random initialization, early training stage, or pre-trained initialization (Renda et al., 2020; You et al., 2020a; Chen et al., 2020c;b;a). Such subnetworks, usually called *winning tickets*, contain much fewer non-zero parameters compared with the original dense networks, but can achieve similar or even better performance when trained in isolation. This discovery undermines the necessity of over-parameterized initialization for successful training and good generalization ability of neural networks (Zhou et al., 2019; Liu et al., 2019b). This new scheme implies the possibility to train a highly compact subnetwork instead of a prohibitively one without compromising performance, potentially drastically reducing the computational cost for network training.

However, the current success of LTH essentially depends on an iterative routine called *Iterative Magnitude-based Pruning (IMP)*, which requires repeated cycles of training networks from scratch, pruning and resetting the remaining parameters. IMP makes it extremely expensive and sometimes unstable to find winning tickets at scale, with large models and large datasets (Frankle et al., 2019). To alleviate this drawback, many efforts have been devoted to finding more efficient alternatives to IMP that can identify sparse trainable subnetworks at random initialization, with little-to-no training (Lee et al., 2019; Wang et al., 2020; Tanaka et al., 2020; Frankle et al., 2020b). Unfortunately, these methods all see some performance gap when compared to the winning tickets found by IMP, with often different structural patterns (Frankle et al., 2020b). Hence, IMP remains to be the *de facto* scheme for lottery ticket finding.

Morcos et al. (2019) found that a winning ticket of one dense network can generalize across datasets and optimizers, beyond the original training setting where it was identified. Their work provided a new perspective of reducing IMP cost – to only find one generic, dataset-independent winning ticket for each backbone model, then transferring and re-training it on various datasets and downstream tasks. Compared to this relevant prior work which studies the transferablity of a winning ticket in the same network architecture, in this paper, we ask **an even bolder question**:

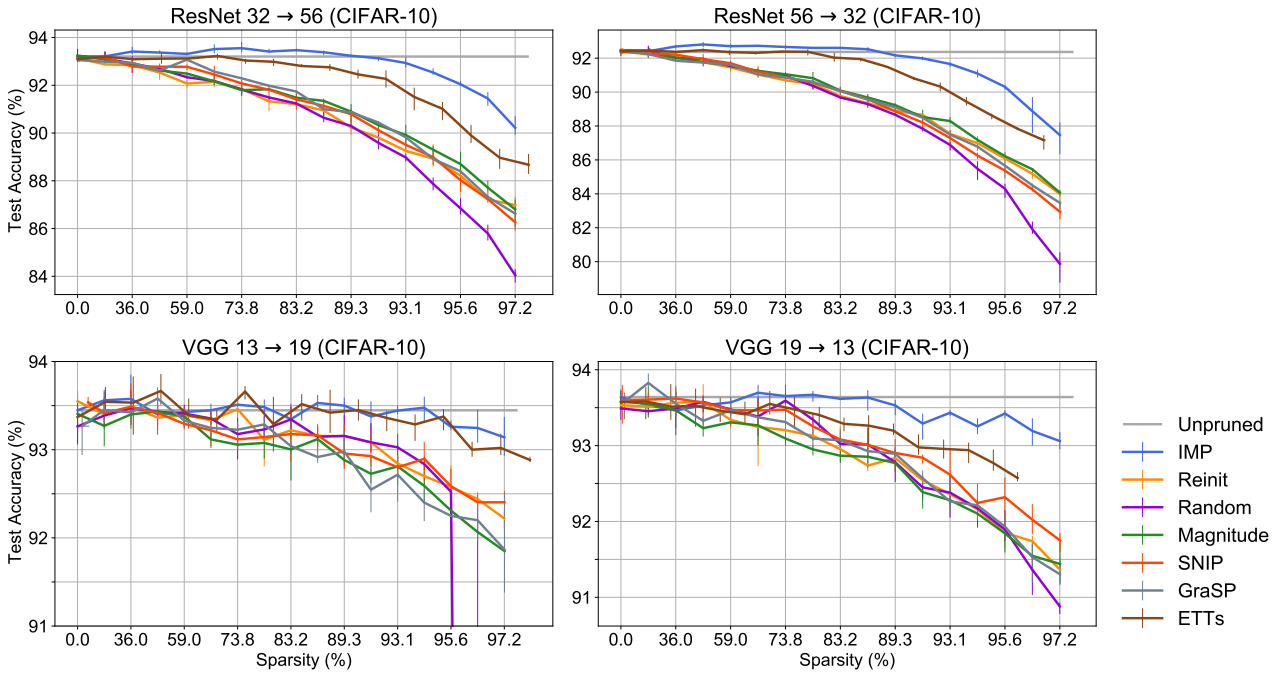


*Figure 1.* Experiments to validate Elastic Lottery Ticket Hypothesis (E-LTH) between two pairs of networks (ResNet-32 and ResNet-56, VGG-13 and VGG-19) trained on the CIFAR-10 dataset. We transform the winning tickets in the source models at different levels of sparsity using the proposed *Elastic Ticket Transformations (ETTs)*, and compare with other pruning methods (please refer to Section 4 for those methods' details), including the state-of-the-art pruning-at-initialization methods (e.g., SNIP and GraSP). Substantial accuracy gaps can be observed between ETTs and other pruning methods, corroborating that ETTs can provide high-quality tickets on the target models that are comparable with the winning tickets found by the expensive IMP. All results are based on three independent runs of experiments.

*Can we transfer the winning ticket found for one network to other different network architectures?*

This question not only has strong practical relevance but also arises theoretical curiosity. On the practicality side, if its answer is yes, then we will perform only expensive IMP for one network and then automatically derive winning tickets for others. It would point to a tantalizing possibility of *once-for-all* lottery ticket finding, and the extraordinary cost of IMP on a "source architecture" is amortized by transferring to a range of "target architectures". A promising application is to first find winning tickets by IMP on a small source architecture and then transfer it to a much bigger target architecture, leading to drastic savings compared to directly performing IMP on the latter. Another use case is to compress a larger winning ticket directly to smaller ones, in order to fit in the resource budgets on different platforms. On the methodology side, this new form of transferability would undoubtedly shed new lights on the possible mechanisms underlying the LTH phenomena by providing another perspective for understanding lottery tickets through their transferablity, and identify shared and transferable patterns that make sparse networks trainable (Gale et al., 2019; Tanaka et al., 2020; Frankle et al., 2020a). Moreover, many deep networks have regular building blocks and repetitive structures, lending to various model-based interpretations such as dynamical systems or unrolled estimation (Greff et al., 2016; Weinan, 2017; Haber et al., 2018; Chang et al., 2018). Our explored empirical methods seem to remind of those explanations too.

### 1.1. Our Contributions

We take the first step to explore how to transfer winning tickets across different architectures. The goal itself would be daunting to impossible, if no constraint is imposed on the architecture differences. Just like general transfer learning, it is natural to hypothesize that two architectures must share some similarity so that their winning tickets may transfer.

In this paper, we focus our initial agenda on network architectures from the same design family (e.g., a series of models via repeating or expanding certain building blocks) but of different depths. For a winning ticket found on one network, we propose various strategies to "stretch" it into a winner ticket for a deeper network of the same family, or "squeeze" it into one for a shallower network. We then compare their performance with tickets directly found on those deeper or shallower networks. We conduct extensive experiments on CIFAR-10 with models from the ResNet and VGG families, and further extend to ImageNet.

From experiments, we seem to reach an affirmative answer to our question in this specific setting. We formalize our observations by articulating the *Elastic Lottery Ticket Hypothesis* (**E-LTH**): by mindfully replicating (or dropping) and re-ordering layers for one network, its corresponding winning ticket could be stretched (or squeezed) into a subnetwork for another deeper (or shallower) network from the same family, whose performance is nearly the same competitive as the latter's winning ticket directly found by IMP. Those stretched or squeezed winning tickets also largely outperform the sparse subnetworks found by pruning-at-initialization approaches (Lee et al., 2019). Besides, we also provide intuitive explanations for the preliminary success.

Lastly, we stress that our method has a pilot-study nature, and is not assumption-free. The assumption that different architectures come from one design "family" might look restrictive. However, we point out many state-of-the-art deep models nowadays are designed in "families", such as ResNets (He et al., 2016), MobileNets (Howard et al., 2017), EfficientNets (Tan & Le, 2019), and Transformers (Vaswani et al.). Hence, while we see significant practicality from the current results, we are also ready to extend them to more general notions - which we discuss in Section 5.

## 2. Related Work

### 2.1. Lottery Ticket Hypothesis Basics

Frankle & Carbin (2019) pointed out the existence of winning tickets at random initialization, and showed that these winning tickets can be found by IMP. Denote $f(x;\theta)$ as a deep network parameterized by $\theta$ and $x$ as its input. A sub-network of $f$ can be characterized by a binary mask $m$, which has exactly the same dimension as $\theta$. When applying the mask $m$ to the network, we obtain the sub-network $f(x;\theta \odot m)$, where $\odot$ is the Hadamard product operator.

For a network initialized with $\theta_0$, the IMP algorithm with rewinding (Frankle et al., 2019; 2020a) works as follows: $(1)$ initialize $m$ as an all-one mask; $(2)$ train $f(x;\theta_0 \odot m)$ for $r$ steps to get $\theta_r$; $(3)$ continue to fully train $f(x;\theta_r \odot m)$ to obtain a well-trained $\theta$; $(3)$ remove a small portion $p \in (0,1)$ of the remaining weights with the smallest magnitudes from $\theta \odot m$ and update $m$; $(5)$ repeat $(3)$-$(4)$ until a certain sparsity ratio is achieved. Note that when $r = 0$, the above algorithm reduces to IMP without rewinding (Frankle & Carbin, 2019). Rewinding is found to be essential for successfully and stably identifying winning tickets in large networks (Frankle et al., 2019; 2020a).

Zhou et al. (2019) investigated different components in LTH and observed super-masks in winning tickets. You et al. (2020a) identified Early-Bird Tickets that contain structured sparsity, which emerge at the early stage of the training process. Morcos et al. (2019); Chen et al. (2020b) studied the transferability of winning tickets between datasets; the former focuses on showing one winning ticket to generalize across datasets and optimizers; and the latter investigates LTH in large pre-trained NLP models, and demonstrates the winning ticket transferability across downstream tasks.

### 2.2. Pruning in the Early Training Stage

A parallel line of works study the pruning of networks at either the initialization or the early stage of training, so that the resulting subnetworks can achieve close performance to the dense model when fully trained. *SNIP* (Lee et al., 2019) proposed to prune weights that are the least salient for the loss in the one-shot manner. *GraSP* (Wang et al., 2020) further exploited the second-order information of the loss at initialization and prune the weights that affect gradient flows least. Tanaka et al. (2020) unified these methods under a newly proposed invariant measure called *synaptic saliency*, and showed that pruning iteratively instead of in one-shot is essential for avoiding layer collapse. The authors then propose an iterative pruning method called *SynFlow* based on synaptic saliency that requires no access to data.

However, it is observed in Frankle et al. (2020b) that vanilla magnitude-based pruning (Han et al., 2015) is as competitive as the above carefully designed pruning methods in most cases, yet all are inferior to IMP by clear margins). The pruning-at-initialization methods such as SNIP and GraSP were suggested to identify no more than a layer-wise pruning ratio configuration. That was because the SNIP/GraSP masks were found to be insensitive to mask shuffling within each layer, while LTH masks would be impacted a lot by the same. Those suggest that existing early-pruning methods are not yet ready to be the drop-in replacement for IMP.

Instead of searching for static sparse subnetworks, another line of works, called *dynamic sparse training (DST)*, focuses on training sparse subnetworks from scratch while dynamically changing the connectivity patterns. DST was first proposed in (Mocanu et al., 2018). Following works improve DST by parameter redistribution (Mostafa & Wang, 2019; Liu et al., 2021a) and gradient-based methods (Dettmers & Zettlemoyer, 2019; Evci et al., 2020). A recent work (Liu et al., 2021b) suggested that successful DST needed to explore the training of possible connections sufficiently.

### 2.3. Network Growing

Broadly related to this paper also includes the research on network growing. Net2Net (Chen et al., 2015) provided growing operations that preserve the functional equivalence for widening and deepening the network. Network Morphism (Wei et al., 2016) further extended Net2Net to more flexible operations that change the network architecture but maintains its functional representation. Network Morphism is used in (Elsken et al., 2017) to generate several neighbour-

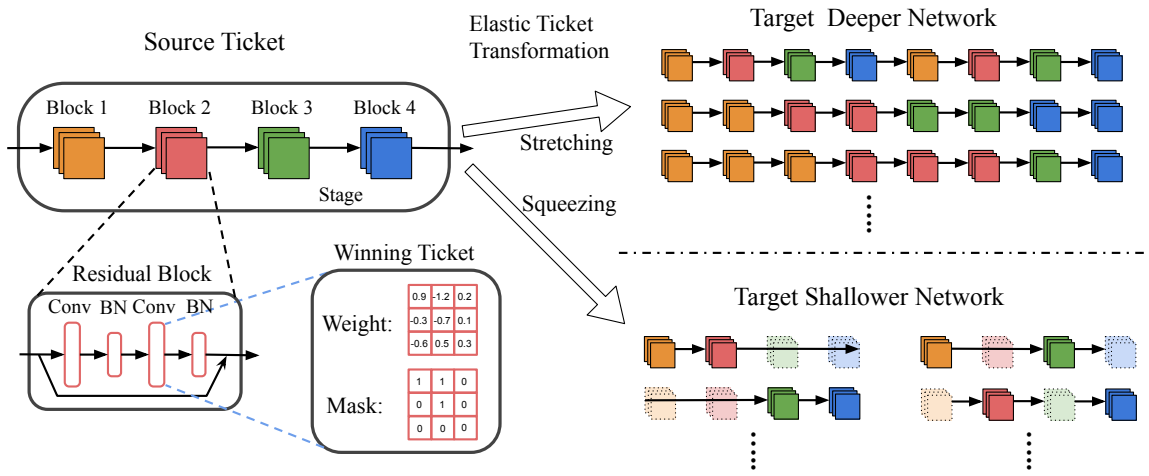


*Figure 2.* An illustration of Elastic Lottery Ticket Hypothesis, using a ResNet-type network as an example, which includes five residual blocks in each stage. Note that we hide "Block 0" who downsamples the input features as we will always preserve this block, without replicating or dropping. On the left, we use IMP to find winning tickets of the source network. On the right, we can either stretch the source ticket to a deeper network, or squeeze it to a shallower network, using the proposed Elastic Ticket Transformation (ETT). In both cases, we have a number of strategies for mindfully replicating, dropping and re-ordering the blocks.

ing networks to select the best one based on some training and evaluation. However, this requires comparing multiple candidate networks simultaneously. A more recent work, FireFly (Wu et al., 2020), proposed a joint optimization framework that grows the network during training.

Despite our similar goal of transferring across architectures, the above methods do not immediately fit into our case of winning tickets. We face the extra key hurdle that we need to transfer (stretch or squeeze) not only general weights, but also the sparse mask structure while preserving approximately the same sparsity. For example, Net2Net and FireFly add (near) identity layers as the initialization for the extra layers, which are already highly sparse and incompatible with IMP without modification. Network Morphism decomposes one layer into two by alternating optimization, which will break the sparse structure of the source winning tickets.

## 3. Elastic Lottery Ticket Hypothesis

The overall framework of E-LTH is illustrated in Figure 2. At the core of E-LTH are a number of *Elastic Ticket Transformations (ETTs)* proposed by us to convert the sparse architecture of a winning ticket to its deeper or shallower variants. We present rules of thumb for better performance, as well as discussions and intuitive explanation for why they do/do not work. For the simplicity of illustration, we will use the ResNet family as examples in this section.

**Terminology:** Suppose we have identified a winning ticket $f^s(\theta_r^s, m^s)$ on a **source network** using IMP, characterized by the rewinding weights $\theta_r^s$ and the binary mask $m_s$, where $r$ is the rewinding steps and $s$ stands for "source". Our goal is to transform the ticket $f^s$ into a **target network** $f^t\left(g(\theta_r^s, m^s), h(\theta_r^s, m^s)\right)$ with a different depth directly, avoiding running the expensive IMP on the target network again. Here, the superscript $t$ stands for "target". $g(\cdot,\cdot)$ and $h(\cdot,\cdot)$ are the transformation mapping for rewinding weights and the mask, respectively, taking the source weights and source mask as inputs.

### 3.1. Stretching into Deeper Tickets

We first present ETTs for stretching a winning ticket, which select certain layers to replicate, both in the (rewinding) weights and the corresponding sparse mask. Below, we discuss the major design choices.

**Minimal unit for replication.** Intuitively, "layer" is the most natural choice for the minimal unit in neural networks. Normalization layers (e.g., Batch Normalization layers (Ioffe & Szegedy, 2015)) are widely used in modern networks and play essential roles in the successful training of the networks. Therefore, in this paper, ETTs considers a linear or convolutional layer and the normalization layer associated with it as a whole "layer". We find this to work well for VGG networks (Simonyan & Zisserman, 2014).

However, in ResNet (He et al., 2016) or its variants, which consist of multiple residual blocks, we consider the residual block as the minimal unit, because they are the minimal repeating structure. Moreover, the residual blocks, which are composed of two or three convolutional layers with normalization layers and a shortcut path, can be interpreted from different perspectives. For example, one residual block is interpreted as one time step for a forward Euler discretization in dynamical systems (Weinan, 2017) and interpreted as one

iteration that refines a latent representation estimation (Greff et al., 2016). More discussion can be found in Section 3.3.

**Invariant components.** To ensure generalizability, we prefer minimal modification to the architectures involved. Hence, we will keep several components of the networks unaffected during stretching:

- **The number of stages**, which are delimited by the occurrences of down-sampling blocks. Adding new stages change the dimensions of intermediate features and create new layers with totally different dimensions, which are hard to be transferred from source tickets.
- **The down-sampling blocks**. In ResNet networks, the first block of each stage is a down-sampling block. In ETTs, we directly transplant the down-sampling blocks to the target network without modification and keep the one-to-one relationship with the stages.
- **The input and output layers**. Thanks to the first invariance, the input and output layers have consistent dimensions in the source and target networks and thus can be directly reused.

**Which units to replicate?** Let us take ResNet-20 (source) and ResNet-32 (target) as an example. Each stage of ResNet-20 contains one down-sampling block, $B_0^s$, and two normal blocks, $B_1^s$ and $B_2^s$, while ResNet-32 has four normal blocks for each stage. To stretch a winning ticket of ResNet-20 to ResNet-32, we need to add two residual blocks in each stage. Then, *should we replicate both $B_1^s$ and $B_2^s$ once, or only replicate $B_1^s$ (or $B_2^s$) twice? If the latter, which one?*

For the first question, in ETTs we choose to replicate more unique source blocks and for fewer times, i.e., we prefer to replicate both $B_1^s$ and $B_2^s$ once in the above example. As will be explained in Section 3.3, this could be understood as a uniform interpolation of the discretization steps in the dynamical system, a natural way to go finer-resolution along time. Another practical motivation for the choice is that such strategy can better preserve the sparsity ratio of each stage and thus the overall network sparsity. If $B_1^s$ and $B_2^s$ have different sparsity ratios (we observe that usually later blocks are sparser), replicating only one of them for several times will either increase or decrease the overall sparsity; in comparison, replicating both $B_1^s$ and $B_2^s$ proportionally maintains the overall sparsity ratio.

If we compare replicating either $B_1^s$ or $B_2^s$, the former will lead to better performance due to resultant lower sparsity (see above explained), and the latter causing lower performance due to effective higher sparsity. Both are understandable and could be viewed as trade-off options.

**How to order the replicated units?** Following the ResNet-20 to ResNet-32 example above, we replicate $B_1^s$ and $B_2^s$ once for each of them, resulting in two possible ordering of the replicated blocks: (1) $B_0^s \to B_1^s \to B_2^s \to B_1^s \to B_2^s$, which we call *appending* as the replicated blocks are appended as a whole after the last replicated block; (2) $B_0^s \to B_1^s \to B_1^s \to B_2^s \to B_2^s$, which we call *interpolation* as each replicated block is inserted right after its source block. We conduct experiments using both ordering strategy and observe comparable performance.

### 3.2. Squeezing into Shallower Tickets

Squeezing the winning tickets in a source network into a shallower network is the reverse process of stretching, and now we need to decide which units to drop. Therefore, we have symmetric design choices as ticket stretching, besides following the same minimal unit and invariances.

To squeeze a winning ticket from ResNet-32 into ResNet-20, we need to drop two blocks in each stage. The first question is: should we drop consecutive blocks, e.g., $B_1^s, B_2^s$ or $B_3^s, B_4^s$, which corresponds to the inverse process of the *appending* ordering, or drop non-consecutive blocks, e.g., $B_1^s, B_3^s$ or $B_2^s, B_4^s$, which corresponds to the inverse process of the *interpolation* ordering above? The second question is: in either case, should we drop the earlier or the later blocks?

According to the extensive ablation study in Section 4.1, we find that ETTs are not sensitive to whether we drop blocks consecutively or at intervals; however, it is critical that we do not drop too many early blocks.

### 3.3. Rationale and Preliminary Hypotheses

We draw two perspectives that may intuitively explain the effectiveness of ETTs during stretching and squeezing. Note that both explanations are fairly restricted to the ResNet family, while E-LTH seems to generalize well beyond ResNets. Hence they are only our preliminary hypotheses, and further theoretical understandings of ETTs will be future work.

**Dynamical systems perspective:** ResNets have been interpreted as a discretization of dynamical systems (Weinan, 2017; Haber et al., 2018; Chang et al., 2018). Each residual block in the network can be seen as one step of a forward Euler discretization, with an implicit time step size of an initial value ordinary differential equation (ODE). Under this interpretation, adding a residual block right after each source block, that copies the weights and batch normalization parameters from the source block, can be seen as a uniform interpolation of this forward Euler discretization, by doubling the number of time steps while halving the implicit step size. Under the same unified view, if we replicate only $B_1^s$ (or $B_2^s$), that could be seen as performing non-uniform interpolation, that super-solves only one time interval but not others. Without pre-assuming which time step is more critical, the uniform interpolation is the plausible choice, hence providing another possible understanding of our design choice in Section 3.1.

*Table 1.* Hyperparameters for network training and IMP, following the settings used in Frankle & Carbin (2019); Frankle et al. (2019). We conduct all CIFAR-10 experiments for three independent runs with different seeds and one run for ImageNet experiments.

| Network | Dataset | Epochs | Batch | Rate | Schedule | Warmup | Rewinding | Unpruned |
|---|---|---|---|---|---|---|---|---|
| ResNet | CIFAR-10 | 160 | 128 | SGD, 0.1 | x0.1 at 80, 160 epoch | - | 1,000 steps | - |
| | ImageNet | 90 | 1,024 | SGD, 0.4 | x0.1 at 30, 60, 80 epoch | 5 epochs | 5 epochs | - |
| VGG | CIFAR-10 | 160 | 128 | SGD, 0.1 | x0.1 at 80, 160 epoch | - | 200 steps | Last linear |

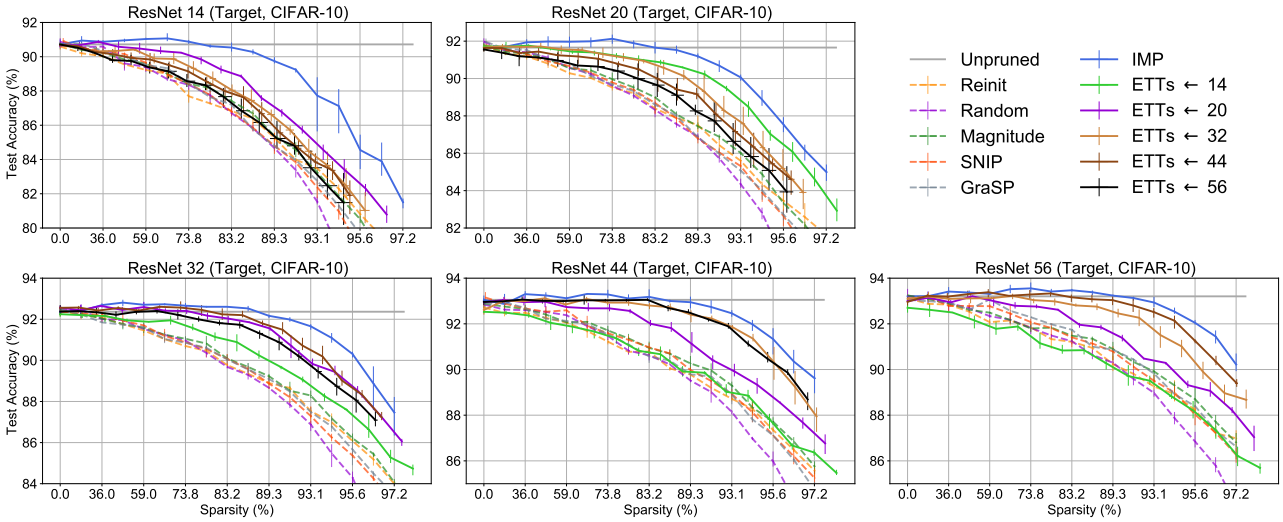


*Figure 3.* Results of the experiments on CIFAR-10 using 5 different ResNet networks: ResNet-14, ResNet-20, ResNet-32, ResNet-44, ResNet-56, left to right, top to bottom. ETTs $\leftarrow n$ in the legend stands for the ticket transformed from ResNet-$n$ using ETTs, which is also absent in the subfigure where ResNet-$n$ is the target network. Curves for pruning methods are dashed.

**Unrolled estimation perspective:** Greff et al. (2016) interprets ResNets as unrolled iterative estimation process: each stage has a latent representation for which the first block in this stage generates a rough estimation and the remaining blocks keep refining it. This perspective provides a strong motivation to keep the first block of each stage untouched, as it contributes to the important initial estimate for the latent representation. Replicating or dropping the remaining residual blocks will incrementally affect the latent representation estimation. The interpolation method for stretching then corresponds to running every refining step for multiple times and the appending method corresponds to re-running (part of) the refining process again for better estimation.

It is also implied by Greff et al. (2016) that since each residual block is incremental, removing blocks only has a mild effect on the final representation, providing intuition for the effectiveness of squeezing tickets by dropping blocks.

## 4. Numerical Evaluations

We conduct extensive experiments on CIFAR-10 and then extend to ImageNet, transferring the winning tickets across multiple models from ResNet family and VGG family. The hyperparameters for the standard training and LTH experiments are shown in Table 1. We follow the official implementation[1] and the hyperparameters of LTH (Frankle & Carbin, 2019; Frankle et al., 2019). Rewinding is used by default for IMP. We run all experiments three times independently with different seeds.

Besides IMP and the unpruned dense models, we will also compare ETTs with state-of-the-art pruning-at-initialization methods, including **SNIP** (Lee et al., 2019), **GraSP** (Wang et al., 2020) and One-shot **Magnitude**-based pruning which was suggested by Frankle et al. (2020b) as a strong baseline. We also include two common pruning baselines in LTH works: (1) *Reinitialization* (**Reinit**), which preserves the identified sparse mask but reinitialize the rewinding weights; and (2) **Random** Pruning, which keeps the rewinding weights yet permuting the masks in a layerwise fashion (e.g., only preserving the layerwise sparsity ratios).

Throughout this paper, we use "sparsity" or "sparsity ratio" to represent the portion of zero elements in networks due to pruning. Thus, the higher the sparsity ratio, the more parameters are pruned. Note that ETTs may cause (slightly) misaligned sparsity ratios with IMP and other pruning methods because replicating and dropping blocks may result in the sparsity changes in a less flexible and controllable way. For a fair comparison, all pruning methods will match

[1] shorturl.at/bnw79

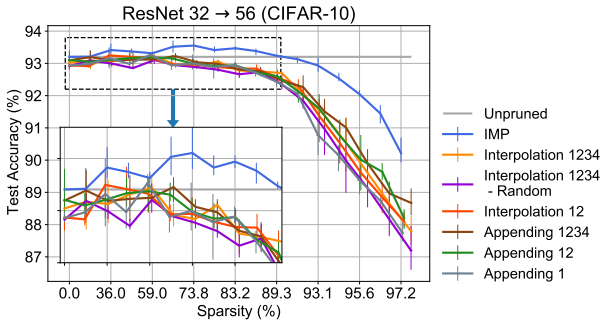


*Figure 4.* Ablation study on replicating unique units on CIFAR-10, using ETTs to transform tickets in ResNet-32 to ResNet-56.

the sparsity ratios of the winning tickets generated by IMP, to the best possible extent. We apply pruning methods to the rewinding weights instead of initialization because it is found in Frankle et al. (2020b) that the former can substantially improve LTH performance.

### 4.1. An Ablation Study on ResNet-32 and Resnet-56

In this subsection, we conduct an ablation study about the rules of thumb for constructing better ETTs as discussed in Section 3. We investigate the selection of replicated or squeezed units when we apply ETTs to stretch or squeeze a winning ticket, and the order of replicated units for the stretching case. All results reported are the average of three independent runs of experiments.

**Should we replicate more unique units?** ResNet-32 has five residual blocks in each stage, notated as $[B_0, \dots B_4]$. As we discussed in Section 3, we leave the downsampling block $B_0$ alone and only play with the rest four. We consider three options, i.e., replicating $(1)$ all four blocks; $(2)$ the first two blocks; $(3)$ and the first block $(B_1)$. We run for each option both the appending and the interpolation methods (note that appending interpolating the first block yields the same resulting model). The results are shown in Figure 4, where the numbers in the legend are the indices of replicated blocks. We can see that replicating four blocks is slightly better than replicating two for both appending and interpolation and much better than only replicating the first by clear gap. The gaps are clearer when the sparsity ratios grow higher. At lower sparsity ratios, the differences between those options are smaller. We also include an option, "Interpolation 1234 - Random", in which we randomly permute the masks of the replicated blocks (but preserves the masks of the original blocks). "Interpolation 1234" is better than its random-pruning variant at all sparsity ratios and the advantage is much evident at high sparsity ratios.

**The earlier or the later units?** The next question in ETTs that follows is: should we replicate earlier units in stretching for better performance, or the later ones? And similarly, should we drop earlier units when squeezing the tickets? Is

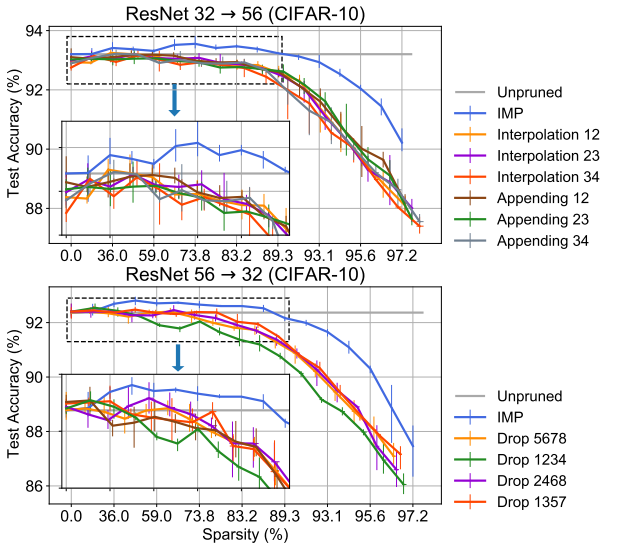


*Figure 5.* Results of the ablation study on replicating/dropping earlier or later units the networks. Top figure: stretching the winning tickets of ResNet-32 into ResNet-56 by appending/interpolating different residual blocks. Bottom figure: squeezing the winning tickets of ResNet-56 into ResNet-32 by dropping four blocks at different positions. Results are the average of three independent runs with error bars hidden for better visualization.

one option always better than the other? For the stretching part, we try to replicate $(B_1, B_2)$, $(B_2, B_3)$, $(B_3, B_4)$ using appending and interpolation methods. Results in the top subfigure of Figure 5 show the advantage of replicating earlier blocks than the later ones with high sparsity, while the differences are less obvious in the low sparsity range. For squeezing, we also try different options of dropping blocks as shown in the bottom subfigure of Figure 5 and find that ETTs is not sensitive to dropping consecutive or non-consecutive blocks, which is consistent to our observation on the stretching experiments, as long as we do not drop earlier blocks too much – we can see a significant decrease in performance when we drop the first four blocks at the same time.

### 4.2. More Experiments on CIFAR-10

**Experiments on more ResNets.** In addition to the transformation between ResNet-32 and ResNet-56 in the ablation study, we apply ETTs that uses *appending* operations to more ResNet networks to transfer the winning tickets across different structures. A more complete set of results is presented in Figure 3. We can see that the tickets generated by ETTs from different source networks clearly outperform other pruning methods, with the only exception of transferring then smallest ResNet-14 to large target networks.

Another (perhaps not so surprising) observation we can draw from Figure 3 is that ETTs usually works better when the source and target networks have a smaller difference in

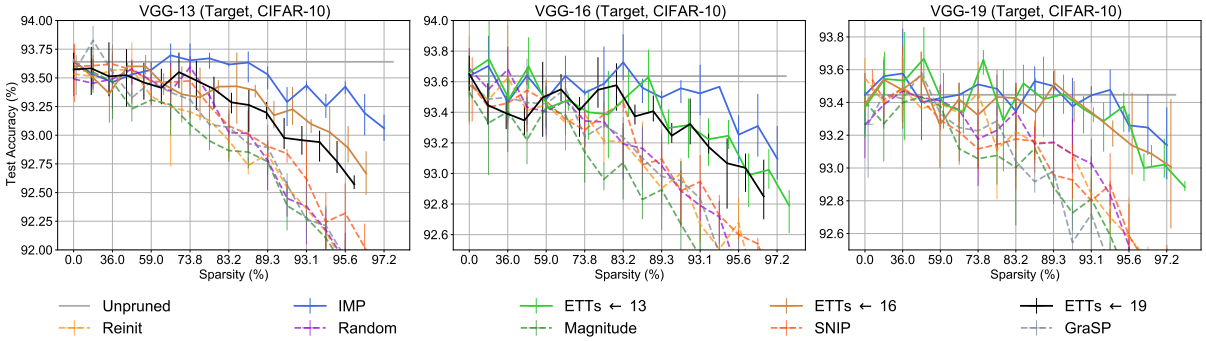


*Figure 6.* Results of the experiments on CIFAR-10 using 3 VGG networks - VGG-13, VGG-16 and VGG-19. ETTs $\leftarrow n$ in the legend stands for the ticket transformed from VGG-$n$ using ETTs, which is also absent in the subfigure where VGG-$n$ is the target network. Curves for pruning methods are dashed.

*Table 2.* Experiments for ResNets on ImageNet.

| Network | ResNet-18 | ResNet-26 | ResNet-34 |
|---|---|---|---|
| Block Config | 2-2-2-2 | 2-3-4-3 | 3-4-6-3 |
| Full Model | 69.96% | 72.56% | 73.77% |
| IMP | 70.22% | 72.94% | 74.20% |
| Reinit | 62.88% | 66.97% | 68.36% |
| Random | 63.10% | 67.07% | 68.68% |
| Magnitude | 64.96% | 68.51% | 69.74% |
| SNIP | 62.23% | 67.51% | 69.35% |
| GraSP | 62.85% | 67.24% | 69.23% |
| ETTs $\leftarrow$ 18 | - | 71.29% | 71.86% |
| ETTs $\leftarrow$ 26 | **68.17%** | - | **73.37%** |
| ETTs $\leftarrow$ 34 | 68.08% | **72.47%** | - |

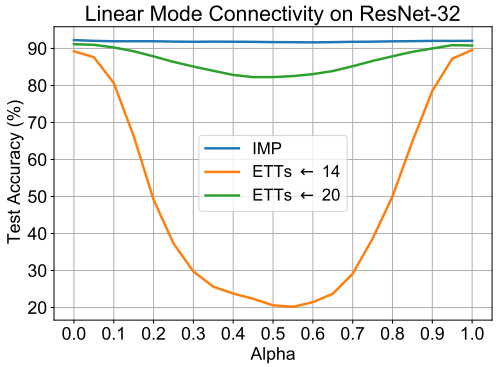


*Figure 7.* Linear mode connectivity property on ResNet-32. We compare winning tickets directly found on ResNet-32 using IMP and E-LTH tickets transferred from on ResNet-l4 and -20.

depth. For example, when ResNet-44 is the target network, ETTs tickets transformed from ResNet-32 and ResNet-56 outperform the tickets transformed from ResNet-14 and ResNet-20 by notable gaps. Similar comparisons can also be drawn on other target networks. On the other hand, the tickets transformed from ResNet-14 have the best performance than other source networks on ResNet-20, and are nearly as competitive as IMP. However, on ResNet-32, the performance of ETTs from ResNet-14 lies between ETTs from other source models and pruning-at-initialization methods. When the target model goes to ResNet-44 or beyond, ETTs from ResNet-14 performs no better than pruning-at-initialization methods. This implies that our replicating or dropping potentially still introduce noise, which may be acceptable within a moderate range, but might become too amplified when replicating or dropping too many times.

**Experiments on VGG networks.** We also apply ETTs to VGG networks, and report another comprehensive suite of experiments of transforming across VGG-13, VGG-19 and VGG-19 networks. Here we directly replicate/drop "layers", instead of residual blocks. The results shown in Figure 6 convey similar messages as the ResNet experiments, where we see very competitive performance with IMP, even at high sparsity ratios, and significant gaps when compared against other pruning counterparts, especially SNIP and GraSP.

### 4.3. Experiments on ImageNet

Next, we extend the experiments to ImageNet. We adopt three models: ResNet-18, ResNet-26, and ResNet-34. We select sparsity ratio $73.79\%$ as the test bed, at which level IMP with rewinding can successfully identify the winning tickets (Frankle et al., 2019).

Our results are presented in Table 2, where we can see ETTs works effectively when transforming tickets to ResNet-26 and ResNet-34, yielding accuracies comparable to the full model and the IMP-found winning ticket, which are much higher than other pruning counterparts. Transforming to ResNet-18 seems slightly more challenging for ETTs, but its superiority over other pruning methods is still solid.

### 4.4. Linear Mode Connectivity of E-LTH

(Frankle et al., 2020a) observed that winning tickets are stably optimized into a linearly connected minimum under different samples of SGD noises (e.g., data order and augmentation), which means that linear interpolations of two independently trained winning tickets have comparable performance. In contrast, linear interpolations for other sparse subnetworks have severely degraded performance. Here, we observe such "linear mode connectivity" property on tickets

*Table 3.* Results of apply E-LTH to FC layers in VGG networks trained on CIFAR-10 and MLP models trained on MNIST. The sparsity ratio is 89.26% for all experiments.

| Source | Target | | |
|---|---|---|---|
| | VGG-13-2 | VGG-13-3 | VGG-13-5 |
| VGG-13-2 | 93.50% | 93.61% | 93.38% |
| VGG-13-3 | 93.50% | 93.62% | 93.59% |
| VGG-13-5 | 93.44% | 93.59% | 93.38% |
| | MLP-2 | MLP-3 | MLP-4 |
| MLP-2 | 98.06% | 97.95% | 97.99% |
| MLP-3 | 97.91% | 97.83% | 97.83% |
| MLP-4 | 97.88% | 97.91% | 98.03% |

transferred from ResNet-14 and ResNet-20 to ResNet-32, shown in Figure 7. We can see that IMP is stable at the linear interpolations of two trained IMP tickets, as observed in (Frankle et al., 2020a). ETTs from ResNet-20 is less stable than IMP in terms of linear mode connectivity, with a maximal 8.72% accuracy drop, but much more stable than ETTs from ResNet-14, with a maximal 69.72% drop. This is consistent with our observation on model accuracies – ETTs has better linear mode connectivity when the source and target networks have a smaller difference in depth, thus having better performance.

### 4.5. E-LTH on Fully-Connected Layers

While the above results show the efficacy of E-LTH on convolutional neural networks, the feasibility of applying E-LTH to fully connected (FC) layers remains elusive because replicating FC layers could possibly create dead neurons. Here, we run two sets of experiments to verify so.

In the first setting, we vary the number of FC layers on top of the convolutional layers in VGG-13. We transfer between VGG-13 with 2, 3 (the default option for VGG networks) and 5 FC layers, denoted by VGG-13-2/3/5. In the second setting, we transfer between MLP (multilayer perceptron) models trained on MNIST, characterized by layer width configurations – MLP-n represents a MLP with layer widths $\{784, 300 \times n, 100, 10\}$, where 784 and 10 correspond to the input and output layer, respectively. All experiments are run with 89.26% sparsity ratio. The results of these two settings are shown in Table 3, where E-LTH yields less than 0.21% accuracy drop in all cases, showing that E-LTH also succeeds on FC layers.

### 4.6. Comparison to Dynamic Sparse Training

Dynamic sparse training (DST) is an uprising direction to train sparse networks from scratch by dynamically changing the connectivity patterns, whileing the overall sparse ratios (thus computation FLOPs) low. However, we emphasize that a crucial difference between E-LTH and DST is that E-LTH is a "one-for-all" method. For example if we run IMP on ResNet-34 once, we then obtain tickets for ResNet-20, ResNet-44 and more "for free" simultaneously, by applying ETTs. In contrast, DST methods have to run on each architecture independently from scratch. Therefore, any overhead of E-LTH is amortized by transferring to many different-depth architectures: that is conceptually similar to LTH's value in pre-training (Chen et al., 2020b;a).

We train ResNet-34 on ImageNet using one recent state-of-the-art DST algorithm, RigL (Evci et al., 2020) (following default 5x training steps setting), at 73.79% sparsity used in our ImageNet experiment. That yields 73.88% accuracy, against 73.37% for ETTs transferred from ResNet-26. FLOPs comparison: 2.77x (finding the mask once on ResNet-26), 0.26x (training ResNet-34 with transferred mask), versus 1.30x (RigL), with all numbers normalized by the FLOPs of one-pass of dense training on ResNet-34. Hence, if the mask found on ResNet-26 could be re-used for more ResNet variants, then the average cost of training FLOPs per model would grow much lower.

*Table 4.* Results of dataset transferring between SVHN and CIFAR-10 datasets. When the source and target model are different, ETTs (from ResNet-20 to ResNet-32) is applied. When only the source and target datasets are different, we directly transfer the winning tickets found by IMP.

| Source | Target | | Accuracy | Baseline (IMP) |
|---|---|---|---|---|
| | Model | Dataset | | |
| RN-20 SVHN | SVHN | RN-20 | 95.95% | 95.95% |
| | | RN-32 | 96.00% | 96.32% |
| | CIFAR10 | RN-20 | 87.21% | 91.07% |
| | | RN-32 | 88.67% | 92.22% |
| RN-20 CIFAR10 | SVHN | RN-20 | 95.44% | 95.95% |
| | | RN-32 | 95.79% | 96.32% |
| | CIFAR10 | RN-20 | 91.07% | 91.07% |
| | | RN-32 | 91.38% | 92.22% |

### 4.7. Transfer Across Both Architectures And Datasets

Previous to this work, the transferability of lottery tickets was studied in (Morcos et al., 2019; Chen et al., 2020b). Here, we also evaluate E-LTH when there are dataset shifts to tentatively investigate the interaction between architecture and dataset transfer. We follow the settings in (Morcos et al., 2019) about dataset transfer. When the source and target model are different, ETTs (from ResNet-20 to ResNet-32) is applied. When only the source and target datasets are different, we directly transfer the winning tickets found by IMP. The baselines accuracies are from the winning tickets

directly found on the target model and dataset using IMP. Results are shown in Table 4, where we can see that either transferring across dataset or architecture only induces performance drop, especially when transferring from simple SVHN to more difficult CIFAR-10. But the drops do not stack when we transfer both simultaneously. For example, compared with the original IMP, transferring ResNet-20 ticket on SVHN to ResNet-32 on SVHN yields 0.32% lower accuracy; transferring ResNet-20 ticket from SVHN to CIFAR-10 yields 3.86% accuracy drop. However, transferring from SVHN to CIFAR-10 AND from ResNet-20 to ResNet-32 simultaneously yields only 3.55% drop.

## 5. Discussion, Limitations, and Future Work

This paper presents the first study of the winning ticket transferability between different network architectures within the same design family. Our experimental observations are summarized by the Elastic Lottery Ticket Hypothesis (E-LTH): a winning ticket found on one network could be stretched or squeezed into a subnetwork of another deeper or shallower network, whose performance is close to that of the latter's winning ticket directly found by IMP. While sometimes there are still a gaps, E-LTH has already outperformed SOTA pruning-at-initialization methods, e.g., SNIP and GraSP, by significant margins. This finding shows the feasibility of transferring at least partial information of the found lottery tickets across architecture, and suggests many brand-new opportunities in practice, such as highly efficient lottery ticket finding for large networks, and adapting a found ticket under resource constraints.

Despite the existing preliminary findings, there is undoubtedly a huge room for E-LTH to improve. As its foremost limitation at present, the current E-LTH only supports a ticket to scale along the network depth dimension. We also make preliminary attempts to extend ETTs to width transformation, but find stretching or squeezing network widths to be much more challenging than depth transformations. Please find some of our results in the Appendix. We conjecture that width-oriented ETTs will need be inspired and derived from a very different set of tools, than the current depth-oriented tools inspired by Section 3.3. We conjecture some promising new tools can be drawn from the study of ultra-wide deep networks (Jacot et al., 2018).

Even further, what is the prospect for E-LTH to go beyond deepening or widening a source ticket (or vice versa)? Can more sophisticated network topology transforms be considered? Answering this question requires a deep reflection on what makes two architectures "similar", i.e., belong to some same design family with transferable patterns. A recent work (You et al., 2020b) shows that many neural network types can be represented by relational graphs with ease, and a graph generator can yield many diverse architectures sharing certain graph measure property (Erdos-Renyi, small-world, etc.). As future work, we plan to leverage their graph generator to systematically explore a design space of neural networks, and see whether there exists elastic winning tickets among many or all of them. We will also continue exploring the theoretical underpinnings of E-LTH.

## References


Chang, B., Meng, L., Haber, E., Tung, F., and Begert, D. Multi-level residual networks from dynamical systems view. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyJS-OgR-.

Chen, T., Goodfellow, I., and Shlens, J. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Carbin, M., and Wang, Z. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. *arXiv preprint arXiv:2012.06908*, 2020a.

Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., and Carbin, M. The lottery ticket hypothesis for pre-trained bert networks, 2020b.

Chen, X., Cheng, Y., Wang, S., Gan, Z., Wang, Z., and Liu, J. Earlybert: Efficient bert training via early-bird lottery tickets. *arXiv preprint arXiv:2101.00063*, 2020c.

Dettmers, T. and Zettlemoyer, L. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

Elsken, T., Metzen, J.-H., and Hutter, F. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*, 2017.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020a.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Pruning neural networks at initialization: Why are we

missing the mark? *arXiv preprint arXiv:2009.08576*, 2020b.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Greff, K., Srivastava, R. K., and Schmidhuber, J. Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*, 2016.

Haber, E., Ruthotto, L., Holtham, E., and Jun, S.-H. Learning across scales—multiscale methods for convolution neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 8580–8589, 2018.

Lee, N., Ajanthan, T., and Torr, P. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1VZqjAcYX.

Liu, Q., Wu, L., and Wang, D. Splitting steepest descent for growing neural architectures. *arXiv preprint arXiv:1910.02366*, 2019a.

Liu, S., Mocanu, D. C., Pei, Y., and Pechenizkiy, M. Selfish sparse rnn training. *arXiv preprint arXiv:2101.09048*, 2021a.

Liu, S., Yin, L., Mocanu, D. C., and Pechenizkiy, M. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. *arXiv preprint arXiv:2102.02887*, 2021b.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019b.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9 (1):1–12, 2018.

Morcos, A., Yu, H., Paganini, M., and Tian, Y. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *NeurIPS*, pp. 4932–4942, 2019.

Mostafa, H. and Wang, X. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pp. 4646–4655. PMLR, 2019.

Renda, A., Frankle, J., and Carbin, M. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgsACVKPH.

Wei, T., Wang, C., Rui, Y., and Chen, C. W. Network morphism. In *International Conference on Machine Learning*, pp. 564–572. PMLR, 2016.

Weinan, E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

Wu, L., Liu, B., Stone, P., and Liu, Q. Firefly neural architecture descent: a general approach for growing neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R. G., Wang, Z., and Lin, Y. Drawing early-bird tickets: Toward more efficient training of deep networks. In *ICLR*, 2020a.

You, J., Leskovec, J., He, K., and Xie, S. Graph structure of neural networks. In *International Conference on Machine Learning*, pp. 10881–10891. PMLR, 2020b.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *NeurIPS*, pp. 3597–3607, 2019.

## A. More results on CIFAR-10 using ResNet Networks

For all results of experiments presented in Section 4.2 on CIFAR-10 using ResNet and VGG networks, we by default "append" residual blocks (for ResNet networks) or convolutional layers (for VGG networks) that replicate earlier layers when we stretch shallower winning tickets into deeper ones, and "drop" later blocks or convolutional layers when we squeeze deeper winning tickets into shallower ones. These schemes are found to work well based on the ablation studies presented in Section 4.1.

Here, we provide experimental results of more schemes of replicating or dropping residual blocks in ResNet networks to make it easier to compare the effectiveness of different transformation schemes. The results are shown in Figure 9 (targeted on ResNet-14), Figure 10 (targeted on ResNet-20), Figure 11 (targeted on ResNet-32), Figure 12 (targeted on ResNet-44) and Figure 13 (targeted on ResNet-56). In stretching experiments where the target networks are deeper than the source networks, we use "Append" and "Inter" to represent the appending and interpolation method described in Section 3 and use the numbers that follow to represent the indices of residual blocks to be replicated. Correspondingly, in squeezing experiments where the target networks are shallower than the source networks, we use the numbers that follow "Drop" to represent the indices of residual blocks to be dropped.

All experiments follow the settings described at the beginning of Section 4. All results reported are based on three independent runs of experiments.

## B. Preliminary Exploration on Changing the Width

Here, we present our preliminary exploration on extending E-LTH to width transformation.

**Expanding the width** We follow the binary splitting idea in (Liu et al., 2019a; Wu et al., 2020) when we expand the width of a winning ticket to transfer it to wider networks, which is shown in Figure 8. The splitting operation can be recursively applied to all layers of a neural network. It is argued in (Liu et al., 2019a) that this simple binary splitting scheme is sufficient when the network is trained to a local optimum. However, when applying it to winning tickets which are identified in the early training stage, this binary splitting method will result in high level of symmetry, causing difficulty in the network training. To solve this, we try two methods to break the symmetry: (1) adding Gaussian noises to the weights (but not to the masks) in the resulting wider tickets; (2) permuting the weight and mask elements in a layerwise manner but keeping the one-to-one relationship between the weights and masks.

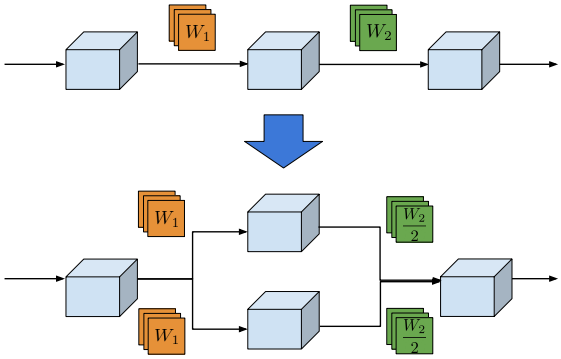


*Figure 8.* Binary splitting scheme to double the width of one layer in neural networks.

We choose the standard ResNet-20 and its variant with doubled width (we call "ResNet-20 Wx2") as the test bed and the results are shown in Table 5. We can see from the results that adding Gaussian perturbations to the replicated weights does help training but still suffers from over $1.6\%$ accuracy gap between the full model and the winning tickets found by IMP, showing the difficult of transforming winning tickets from narrow to wide networks.

**Shrinking the width** We consider shrinking the width of winning tickets as the inverse operation of the above binary splitting scheme. Similar to directly dropping residual blocks or convolutional layers in depth experiments, we first consider directly dropping half of the winning tickets, including both weights and masks. Using ResNet-20 Wx2 and ResNet-20 as our test bed at the same sparsity level $89.26\%$, the tickets transformed from ResNet-20 Wx2 suffer from $3.06\%$ accuracy gap when compared with winning tickets found by IMP in ResNet-20 ($87.73\%$ v.s. $90.79\%$).

*Table 5.* Experiments of expanding the width of winning tickets in ResNet-20. The investigated winning tickets are taken at the sparsity ratio $89.26\%$.

| | ResNet-20 Wx2 |
|---|---|
| Full | 93.71 |
| IMP | 93.76 |
| | ResNet-20 $\rightarrow$ ResNet-20 Wx2 |
| Replicate | 91.92 |
| Gaussian | 92.12 |
| Permute | 91.56 |

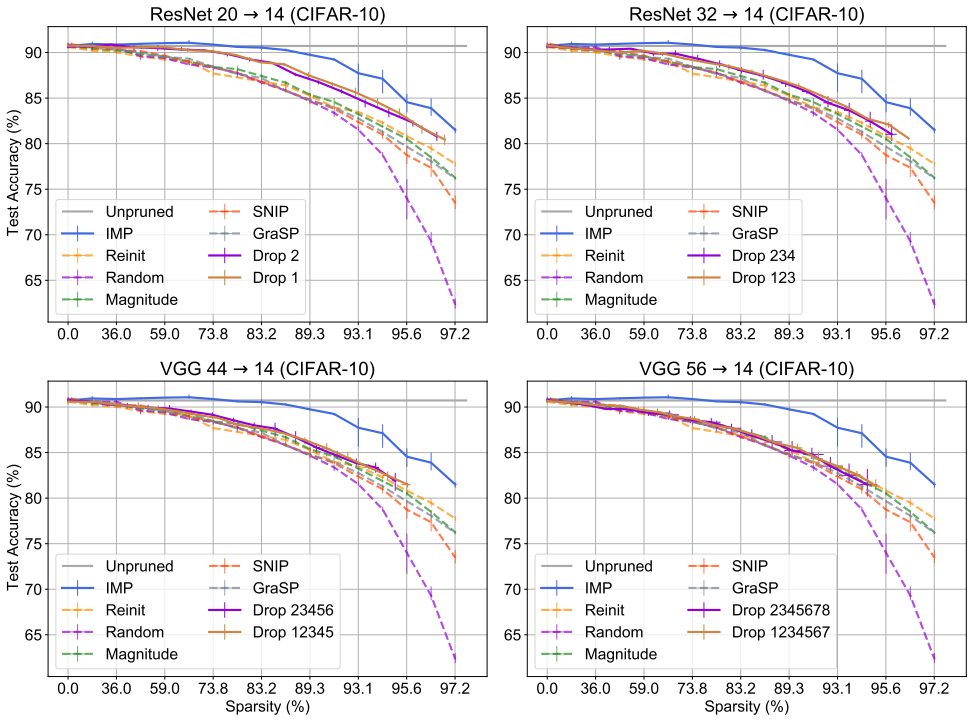


*Figure 9.* Results of the experiments where we transform winning tickets in other networks to ResNet-14 on CIFAR-10. The titles of sub-figures suggest source network. Curves of the average accuracies and the error bars are calculated based on three independent runs of experiments. Curves for pruning methods are dashed.

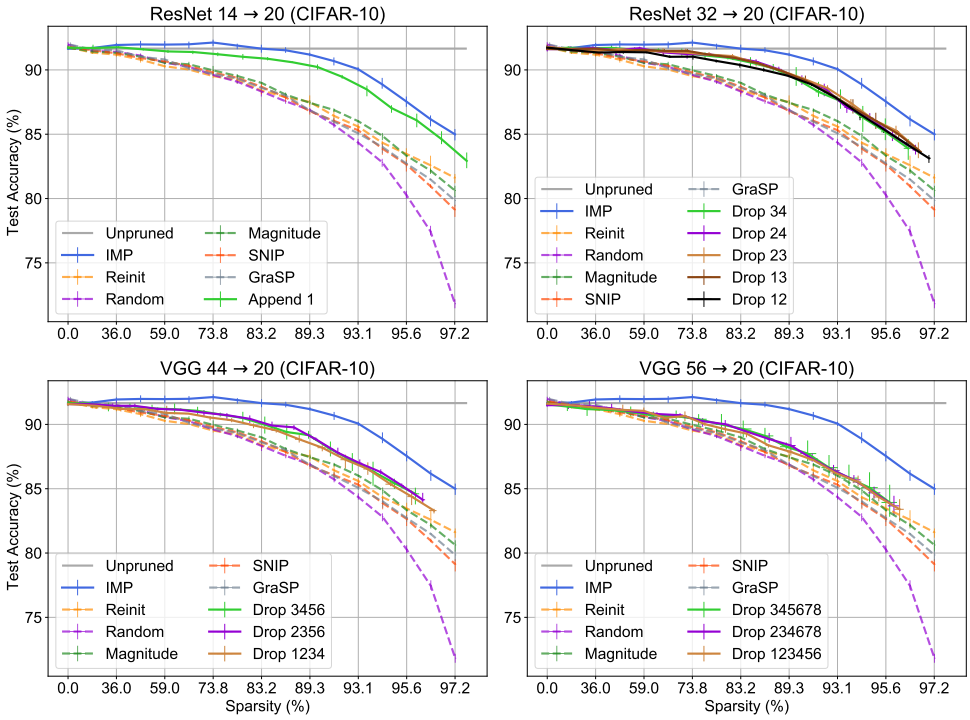


*Figure 10.* Results of the experiments where we transform winning tickets in other networks to ResNet-20 on CIFAR-10. The titles of sub-figures suggest source network. Curves of the average accuracies and the error bars are calculated based on three independent runs of experiments. Curves for pruning methods are dashed.

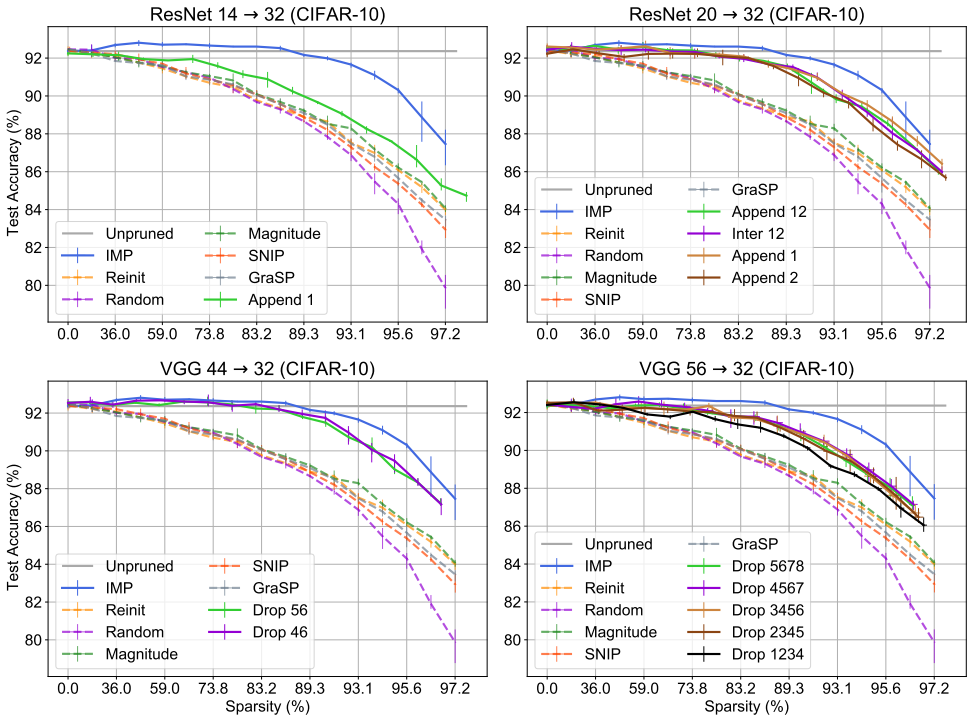


*Figure 11.* Results of the experiments where we transform winning tickets in other networks to ResNet-32 on CIFAR-10. The titles of sub-figures suggest source network. Curves of the average accuracies and the error bars are calculated based on three independent runs of experiments. Curves for pruning methods are dashed.

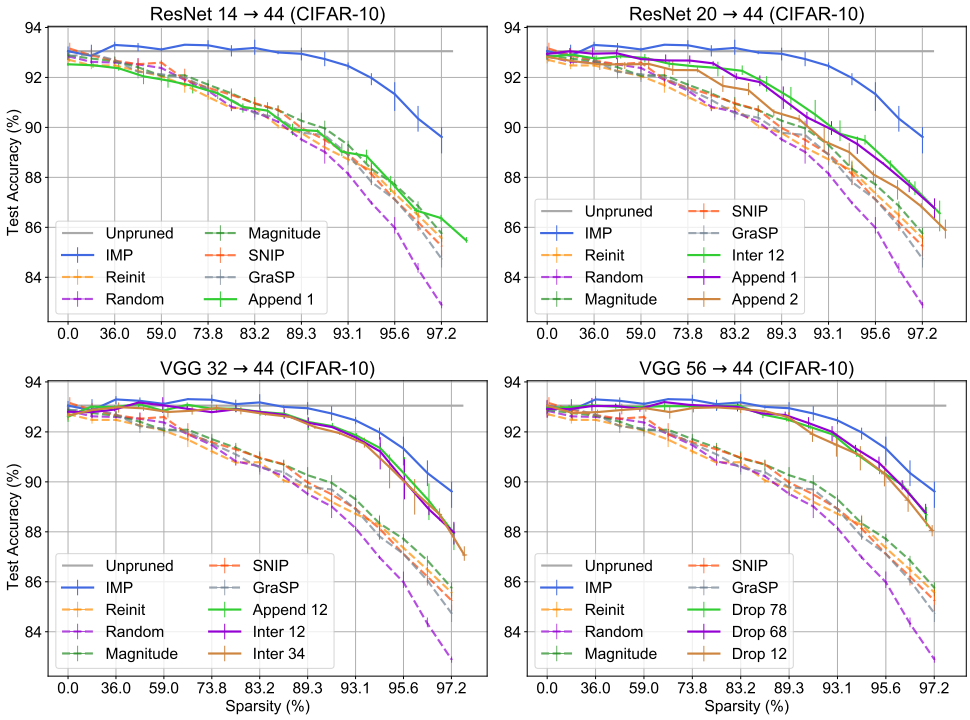


*Figure 12.* Results of the experiments where we transform winning tickets in other networks to ResNet-44 on CIFAR-10. The titles of sub-figures suggest source network. Curves of the average accuracies and the error bars are calculated based on three independent runs of experiments. Curves for pruning methods are dashed.

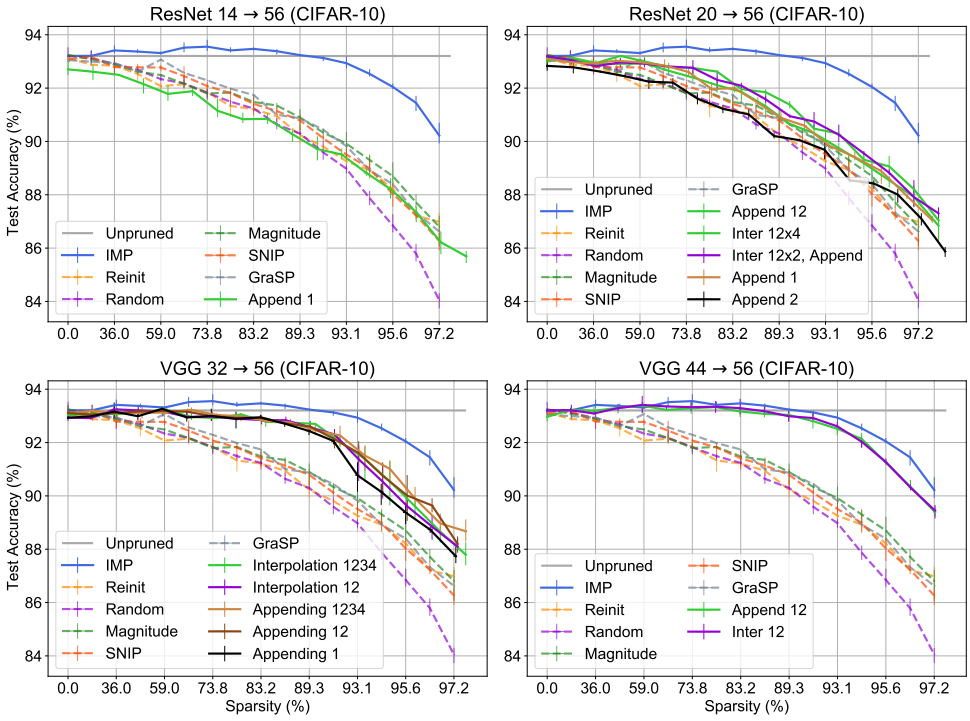


*Figure 13.* Results of the experiments where we transform winning tickets in other networks to ResNet-56 on CIFAR-10. The titles of sub-figures suggest source network. Curves of the average accuracies and the error bars are calculated based on three independent runs of experiments. Curves for pruning methods are dashed. Note: in the experiment of stretching winning tickets in ResNet-20 to ResNet-56, "Inter 12x4" stands for interpolating the blocks $B_1$ and $B_2$ for four times, resulting in $B_1 \to B_1 \to B_1 \to B_1 \to B_2 \to B_2 \to B_2 \to B_2$, and "Inter 12x2, Append" stands for interpolating the blocks $B_1$ and $B_2$ twice and append the resulting four blocks, resulting in $B_1 \to B_1 \to B_2 \to B_2 \to B_1 \to B_1 \to B_2 \to B_2$.